

## APPENDIX A

### Determining Scanned Image First Axis Centerpoint of Reference

```

int yellow_r=200,yellow_g=200,yellow_b1=50,yellow_b2=200;
x=0;
ReTryHoriz:
x+=2;
smallc=bmp.bmHeight,bigc=0;
for(y=bmp.bmHeight/8;y<bmp.bmHeight-bmp.bmHeight/8;y++)
{
    GiveMeSome();
    c=GetPixel(mDC,x,y);
    if(c!=white && GetRValue(c)>yellow_r && GetGValue(c)>yellow_g && GetBValue(c)>yellow_b1 &&
    GetBValue(c)<yellow_b2)
    {
        if(y<smallc)
            smallc=y;
        if(y>bigc)
            bigc=y;
    }
}
if(smallc==bmp.bmHeight && bigc==0 && x<100)
{
    yellow_r-=5;
    yellow_g-=5;
    yellow_b1-=3;
    yellow_b2+=3;
    goto ReTryHoriz;
}
if(smallc==bmp.bmHeight && bigc==0)
{
    char str[100];
    SelectObject(mDC,old);
    DeleteObject(MyBmp);
    DeleteDC(mDC);
    x=bmp.bmWidth;
    y=bmp.bmHeight;
    sprintf(str,"Can't find horiz grid, please check prescan area(%lu,%lu).",x,y);
    MessageBox(hDlg,str,maintitle,MB_OK | MB_ICONSTOP);
    PostQuitMessage(FALSE);
    EndDialog(hDlg,FALSE);
    return(FALSE);
}
gridy=smallc+(bigc-smallc)/2;

```

### Determining Scanned Image Second Axis Centerpoint of Reference

```

int yellow_r=200,yellow_g=200,yellow_b1=50,yellow_b2=200;
y=0;
ReTryVert:
y+=2;
smallc=bmp.bmWidth,bigc=0;
for(x=bmp.bmWidth/8;x<bmp.bmWidth-bmp.bmWidth/8;x++)
{
    GiveMeSome();
    c=GetPixel(mDC,x,y);
    if(c!=white && GetRValue(c)>yellow_r && GetGValue(c)>yellow_g && GetBValue(c)>yellow_b1 &&
    GetBValue(c)<yellow_b2)
    {
        if(x<smallc)
            smallc=x;
        if(x>bigc)
            bigc=x;
    }
}
if(smallc==bmp.bmWidth && bigc==0 && y<100)

```

```

    {
        yellow_r-=5;
        yellow_g-=5;
        yellow_b1-=3;
        yellow_b2+=3;
        goto ReTryVert;
    }
if(smallc==bmp.bmWidth && bigc==0)
{
    char str[200];
    SelectObject(mDC,old);
    DeleteObject(MyBmp);
    DeleteDC(mDC);
    x=bmp.bmWidth;
    y=bmp.bmHeight;
    sprintf(str,"Can't find vertical grid, please check prescan area(%lu-%u,%u)",gridy,x,y);
    MessageBox(hDlg,str,maintitle,MB_OK | MB_ICONSTOP);
    PostQuitMessage(FALSE);
    EndDialog(hDlg,FALSE);
    return(FALSE);
}
gridx=smallc+(bigc-smallc)/2;

```

#### Determining A Starting Radius

```

BOOL bluecheck(COLORREF c)
{
    if(((unsigned char)GetRValue(c)<(unsigned char)185 && (unsigned char)GetGValue(c)<(unsigned
char)185 && (unsigned char)GetBValue(c)>(unsigned char)50)
        return(TRUE);
    return(FALSE);
}
if(bmp.bmHeight>bmp.bmWidth)
{
    if(bmp.bmHeight>3300) // 600dpi, 5.5 inches
        bmp.bmHeight=bmp.bmWidth;
    g=bmp.bmHeight;
}
else
{
    if(bmp.bmWidth>3300) // 600dpi, 5.5 inches
        bmp.bmWidth=bmp.bmHeight;
    g=bmp.bmWidth;
}
bigc2=bigc=0;
smallc2=smallc=g;

//////////
startmeoff=g;
for(x=0;x<512;x+=128)
{
    double r;
    long int myx,myy;
    GiveMeSome();
    c=white;
    for(r=100;r<g && (c==white || !bluecheck(c));r++)
    {
        myx=(int)((double)gridx+(double)MySin(x)*r);
        myy=(int)((double)gridy+(double)MyCos(x)*r);
        c=GetPixel(mDC,myx,myy);
        GiveMeSome();
    }
    GiveMeSome();
    if(r<startmeoff)
        startmeoff=r;
}
if(startmeoff>=g)

```

```

        startmeoff=100;
else
{
    startmeoff=(long) fabs(startmeoff/3);
    if (startmeoff<100)
        startmeoff=100;
}

```

#### Centering A Scanned Image Shape

```

for (x=0;x<MYPOINTS+1;x++)
{
    double r;
    long int myx,myy;
    double rads=x;
    c=white;
for (r=startmeoff;r<g && (c==white || !bluecheck(c));r++) // our radius
{
    GiveMeSome();
    if (reverseclick==0)
        myx=(int)((double)gridx-(double)MySin(rads)*r);
    else
        myx=(int)((double)gridx+(double)MySin(rads)*r);
    myy=(int)((double)gridy+(double)MyCos(rads)*r);
    c=GetPixel(mDC,myx,myy);
}
if (r>=g) // means it's the max value..
{
    char str[200];
    SelectObject(mDC,old);
    DeleteObject(MyBmp);
    DeleteDC(mDC);
    sprintf(str,"No tracing found(%lu), please verify that your prescan region is correct and that the
tracing was done using an approved e.lens pen.",x);
    MessageBox(hDlg,str,maintitle,MB_OK | MB_ICONSTOP);
    PostQuitMessage(FALSE);
    EndDialog(hDlg,FALSE);
    return(FALSE);
}
if (myx>bigc)
    bigc=myx;
if (myx<smallc)
    smallc=myx;
if (myy>bigc2)
    bigc2=myy;
if (myy<smallc2)
    smallc2=myy;
    sprintf(instr,"%u",40+30*x/MYPOINTS);
    SetDlgItemText(hDlg,BAR_CLICK,instring);
    InvalidateRgn(GetDlgItem(hDlg,BAR_CLICK),NULL,TRUE);
    GiveMeSome();
}
}
gridx=smallc+(bigc-smallc)/2;
gridy=smallc2+(bigc2-smallc2)/2;

```

#### Determining A Scanned Image Radial Shape

```

elens.jobdata_datasize=0;
for (x=0;x<MYPOINTS+1;x++)
{
    double r;
    long int myx,myy;
    double rads=x;
    c=white;

```

```

for(r=startmeoff;r<g && (c==white || !bluecheck(c));r++)
{
    GiveMeSome();
    if(reverseclick==0)
myx=(int)((double)gridx-(double)MySin(rads)*r);
    else
myx=(int)((double)gridx+(double)MySin(rads)*r);
myy=(int)((double)gridy+(double)MyCos(rads)*r);
    c=GetPixel(mDC,myx,myy);
}
myx=(int)r;
if(myx<1)
    myx=1;
elens.jobdata_datasize++;

```

#### Determining A Scanned Image Radial Size

```

for(x=0;x<MYPPOINTS+1;x++){
elens.shape[x]=(((double)myx/(double)config.resolution)/(double)0.039370)*(double)100;
elens.shape[x]=(unsigned short int)((unsigned short int)((double)elens.shape[x]/(double)10) * 10L
    +config.calibrate);
}

```

#### Smoothing A Scanned Image Radial Shape

```

for(x=0;x<MYPPOINTS+1;x++){
if(config.makesmooth==1 && x>2)
{
    if(elens.shape[x]>=elens.shape[x-2])
    {
        if(elens.shape[x-1]<elens.shape[x-2] || elens.shape[x-1]>elens.shape[x])
            elens.shape[x-1]=(short int)(elens.shape[x-2]+(elens.shape[x]-elens.shape[x-2])/2);
    }
}
}

```

#### Modify Size of Derived Radial Shape

```

short int y;
short int num=1;
double val=CalcCircum();
do
{
    for(y=0;y<513;y++)
    {
        if(increasesize==1)
            elens.shape[y]=(short int)(original[y]+offset+num);
        else
            elens.shape[y]=(short int)(original[y]+offset-num);
    }
    ShapeToJob();
    JobToShape();
    num++;
} while((increasesize==1 && val>=CalcCircum()) ||
    (increasesize==0 && val<=CalcCircum()));
if(increasesize==1)
    offset=(short int)(offset+(num-1));
else
    offset=(short int)(offset-(num-1));

```

# Rotating A Derived Radial and Smoothed Shape

```
if (nasalup==1)
{
    short int a=original[0];
    for(y=0;y<elens.jobdata_datasize-1;y++)
    {
        short int b;
        b=original[y+1];
        original[y+1]=a;
        a=b;
    }
    original[0]=a;
    rotated--;
}
else
{
    short int a=original[0];
    for(y=0;y<elens.jobdata_datasize-1;y++)
        original[y]=original[y+1];
    original[y]=a;
    rotated++;
}
```